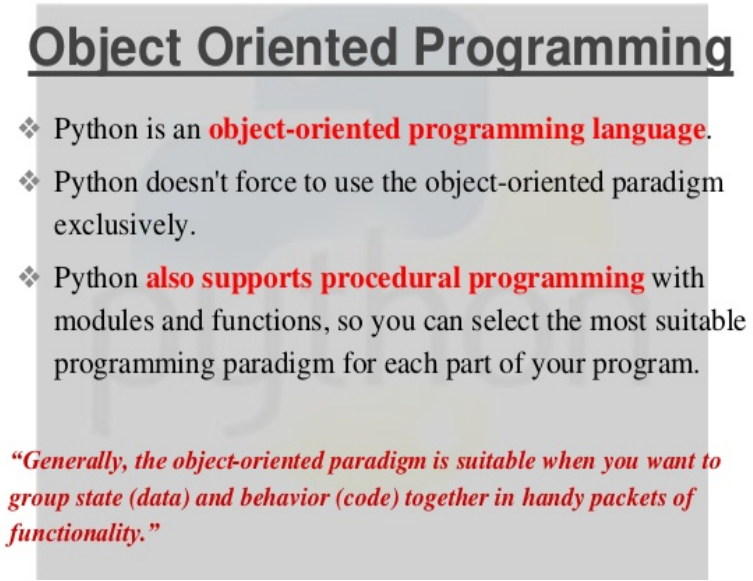
Date=31/07/2020
Lecture By=Shubham Joshi
Subject ⇒ Oops

IN PREVIOUS LECTURE **(QUICK RECAP) Date-30/07/2020**	**In Today's Lecture (Overview)**
Question solution session	Oops in python Class In python Questions for self practice

---

Oops in python

-Oops Stands For Object-oriented programming System
-it is based on the concept of “objects” that contain data and methods.
-Object oriented programming brings together data and its behaviour(methods) in a single location(object) makes it easier to understand how a program works.



“Click here” to know more about it

For video tutorial “click here”

Class In python

-A Class is like an object constructor, or a "blueprint" for creating objects.
Create a Class
To create a class, use the keyword class:

Example
Create a class named MyClass, with a property named x:

```
class MyClass:
  x = 5
```

Create Object
Now we can use the class named MyClass to create objects:

Example
Create an object named p1, and print the value of x:

```
p1 = MyClass()
print(p1.x)
```

The examples above are classes and objects in their simplest form, and are not really useful in real life applications.

To understand the meaning of classes we have to understand the built-in __init__() function.

All classes have a function called __init__(), which is always executed when the class is being initiated.

Use the __init__() function to assign values to object properties, or other operations that are necessary to do when the object is being created:

Example

Create a class named Person, use the __init__() function to assign values for name and age:

```
class Person:
  def __init__(self, name, age):
    self.name = name
    self.age = age

p1 = Person("John", 36)

print(p1.name)
print(p1.age)
```

Note: The `__init__()` function is called automatically every time the class is being used to create a new object.

Object Methods

Objects can also contain methods. Methods in objects are functions that belong to the object.

Let us create a method in the Person class:

Example

Insert a function that prints a greeting, and execute it on the p1 object:

```
class Person:

  def __init__(self, name, age):
```

```
    self.name = name
    self.age = age


  def myfunc(self):
    print("Hello my name is " + self.name)


p1 = Person("John", 36)
p1.myfunc()
```

The self Parameter

The self parameter is a reference to the current instance of the class, and is used to access variables that belongs to the class.

It does not have to be named self , you can call it whatever you like, but it has to be the first parameter of any function in the class:

Example

Use the words *mysillyobject* and *abc* instead of *self*:

```
class Person:

  def __init__(mysillyobject, name, age):
    mysillyobject.name = name
    mysillyobject.age = age
```

```
  def myfunc(abc):

    print("Hello my name is " + abc.name)



p1 = Person("John", 36)

p1.myfunc()
```

“**Click Here**” to Know More About It

For Video Tutorial “**click Here**”

Questions for self practice

Q1. https://leetcode.com/problems/shuffle-string/

Q2. Practice the code taught in the class and add some more attributes in it and methods. And write each and every concept as a comment in the code.